UniVista UI Prototype
Edward Peterlin
Tuesday, March 9, 1999

The prototype interface for UniVista is split up into four different modules. One module consists of simple dialogs needed to prompt the user to perform auxiliary actions, such as specifying the database connection and choosing an initial portion of the program to run. One module allows the user to add in the variables and defaults for an initial addition or change to the version of a code[1]. One module allows the user to create and edit studies[2]. Another module allows the user to run the studies.

Fundamentals

There are certain fundamental terminologies that are used throughout this document regarding interface elements, concepts, and user levels.

UniVista is primarily designed to provide a graphical user interface building tool for physical simulation packages. The division between the description of the simulation program, the interface, and the use of the interface to run the program is split into the UniVista concepts of codes, studies, and runs.

A *code* contains all of the information describing a particular version of a simulation program. This includes descriptions of all of the variables (including default values, short documentation, URL for further documentation, consistency checks, format, and preferred editor), the input and output file formats, and commands that need to be run on a remote machine to invoke the program.

A *study* contains a user interface design for working with a particular code. The study consists of a group of *screens*. Each screen is a window that will eventually be presented to the people using the study to input and examine variables. The screens contain variables, graphical and static text elements, and links to other screens. A study can also have overrides for the variable defaults and short documentation which supercede the code defaults, but only for that study.

A *run* is what corresponds to an invocation of the user interface defined by a study to generate files and execute the simulation software.

There are also different user levels within UniVista for defining these different types of objects. The user level is set on a database granularity, that is, users can have different capabilities depending on which database they connect to.

The *code author* is the person who has the ability to add a new code to UniVista. This person is adding the ability for UniVista to work with a new simulation package, or a new version of an existing one. This person should be familiar with how the simulation package usually interacts with its input files and should be able to describe the variables of the simulation program. This user is defining how UniVista interacts with outside programs.

The *study author* has the ability to create and modify studies within UniVista. They are limited

1. A code is equivalent to defining a particular simulation program to run and describing its input variables and file formats. Different versions of the simulator that have differing input formats or behaviors belong in different codes.
2. A study corresponds to a particular way that a user may wish to use the simulation program. It contains user set default values for variables, perhaps custom descriptions of the task the study is used to solve, and graphical screens of the variables specific to the study.

to using pre-defined codes to create interfaces for. The study author can set new defaults and documentation for variables that is valid within that study only. They can also control the screen appearance and screen hierarchy of the study.

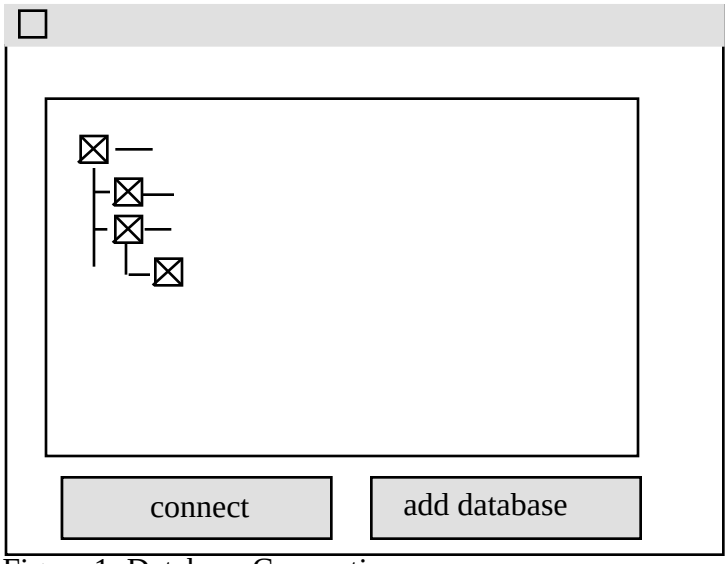

Figure 1: Database Connection

The *run author* is limited to using pre-existing studies to input specific information about the variables. These people can create runs from studies, or they can create runs from using the initial values of variables from a previous run.

This document also uses some terminology for describing common user interface controls. A *tree control* is a hierarchically arranged tree of items. Different levels can be opened and closed by clicking on boxes to the right of each level of the tree. This is similar to the tree control found in the Windows Explorer or the twist down lists for folders in the Macintosh Finder. Drag and drop is used extensively both between windows, but also within a tree control in many places. By dragging an item into a new level of the tree, it can be moved in the hierarchy.

Starting Up

The first thing that a user needs to do when starting UniVista is connect to a database that contains the data files for UniVista. This is done by selecting the database server from the list of databases as shown in figure 1 and clicking “Connect.” A new database server can be added to the list through add database, which brings up a window where the user can type in the address of the server. The control present is a tree since in future versions it is expected for some type of name server to be available to list all of the accessible UniVista database servers within a particular high level domain such as “pppl.gov”. In the first version, however, the user will simply be able to create folders to categorize the servers they want to use (no auto-searching).

After a user connects to the database, they will be prompted for their login and password for that database. After the login phase is completed successfully, UniVista will load the user’s privileges (user level) for that database server, indicating whether they can create or modify certain screens of the study.

After the user logs in, they will next be presented with the main menu. The main menu is simply a group of buttons that allows the user to choose what they want to do first. The user can switch between any of the modes that they have privileges to do at any time from the main menu bar, so the initial choice is non-binding.

The choices are mostly explained in the later sections, with the exception of the program settings. This button is used to allow the user to change preferences of UniVista such as default servers, automatic connections, and user/password saving.

Creating a New Code

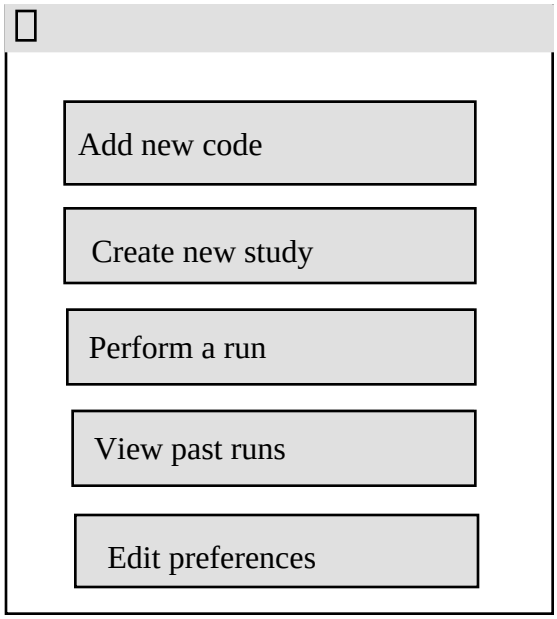

Figure 2: Main Menu

When a new code is created, the user must input the variables and other information needed by UniVista to appropriately generate the output files and run the actual program. When a new code is created, the user is presented with a dialog like in figure 3. In this dialog, the user can see a list of the variables on the left and the properties of a selected variable on the right. In the buttons underneath the tree list of variables, the user can create a new variable grouping or a new variable. Variable groupings are like folders; they help the user to organize the large number of variables for a particular code. On the right hand side of the screen is a properties area with buttons that allows the user to change the default short description of the variable, the variable name, the URL location of extra documentation, default values of the variable, and any consistency checks that need to be done for it. Underneath the Properties area are two buttons that allow the user to commit any changes they have made to the variable or to revert to the old values.

The variables are listed in a tree on the left that allows the transfer of a variable into a new group by dragging its name into one of the group headings within the tree.

In the menu bar for the code editor will be an Import command to allow the user to import a formatted enamelist of values to use. Under the menu bar a Find command will also be located to allow a variable to be found by name or by description contents. These two commands will appear in a consistent place as they are shared by many of the places of the program that use variables.

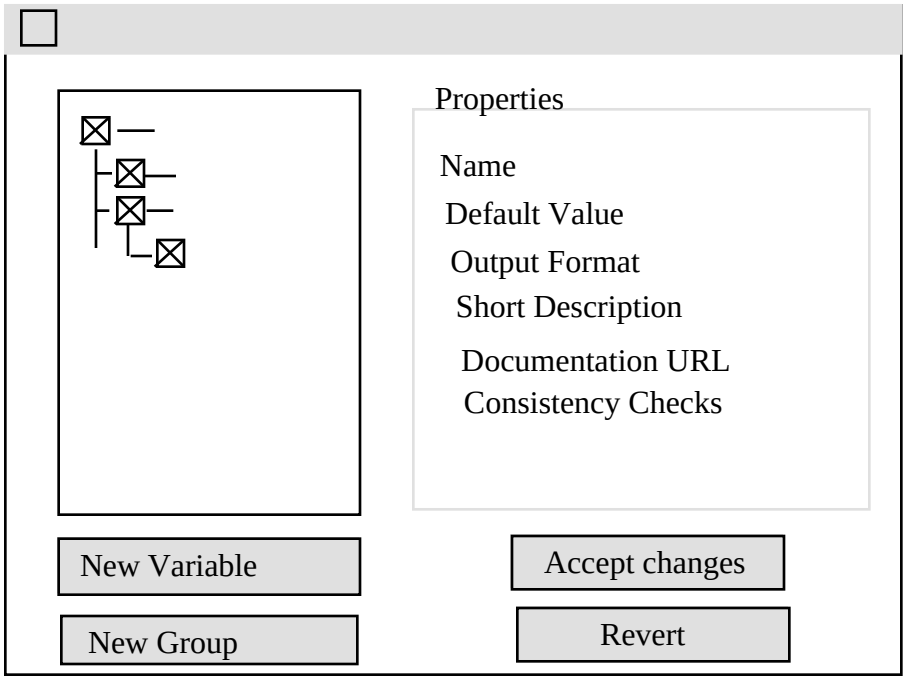

Figure 3: Code Editor

After the variables have been completely specified, the user can hit the “finish code” button and UniVista will create all of the appropriate tables in the database. After this creation, UniVista will then create a default list of screens. Each group will have its own screens with buttons linking to any groups that it contains. On the screen will be a standard layout of the variables that are contained in this group. After these screens are created and in the database, the

study editor component is opened on this default study to allow the user to customize its appearance. The default study is created using the verbose layout engine[3].

The Study Editor

The Study Editor is the component of UniVista that is used to create the graphical user interfaces for particular codes. It is highly graphical in its own nature, allowing users the flexibility to create multiple styles of studies.

There will be a design area which reflects exactly what the run user will see when he creates a new run from the study. Inside of this design area the variables and other objects will be present. When one of these objects is selected, information about it can be changed using the Property Editor (see below). It can also be dragged around inside of the design area to reposition it on the screen. When it is selected, standard Edit menu commands can also be performed on it to copy, cut, or delete it. The design area will also feature an auto-snap and auto-grid feature. *Auto-snap* means that when a variable's name and input area are dragged close to a boundary of second variable, the variable being moved will align itself to the edge of the second variable at a preset distance away. This allows for easy relative arrangement of variables and objects. *Auto-grid* means that objects can only be placed at the intersection points of an invisible grid of evenly spaced lines. The user can have control over the width of this grid. Having no auto-grid snap is equivalent of allowing objects to be placed at the intersection point of a grid where there is 0 spacing between the grid lines.

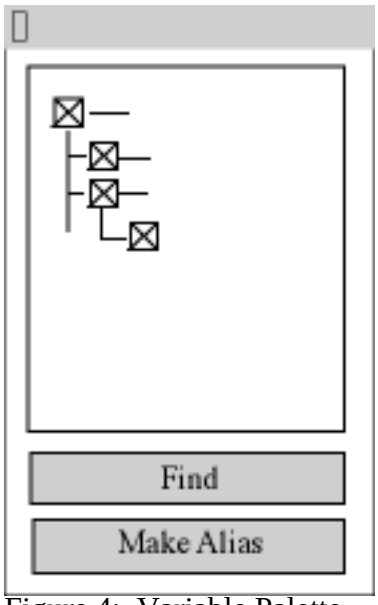

Figure 4: Variable Palette

In addition to the design area, there will be several palettes that float above it. The *variable list palette* (see Figure 4) shows all of the variables that are defined in the code. The tree contains the variables in the group listing as the code author defined. This listing is not mutable. To find a specific variable in the group tree the user can click on the Find button to search for a variable based upon its name or its appearance. The button below it, "Make Alias", is enabled only for variables of array types. This allows a study editor to create an alias of a particular row, column, or entry of an array. This is placed under a special "Aliases" group at the top level of the tree list. The study author can then use this alias variable to place a UI element on the screen for that particular subset of the array only. (caveat...aliasing may not appear in the initial version of UniVista). When a variable in the list is clicked, its default values and default one-liner can be modified using the Property Editor. To add a variable to a particular screen, the user simply drags the variable name from this list into the design area and the variable will be added. If a list of variables is dragged to the entry area, they will be added using the terse layout engine. In the list of variables, variables that have any properties from the code overridden will appear in red, variables that have been placed on a screen in the study will appear as normal, and variables that have not yet been placed on any screen will appear in yellow.

There is a second palette, the *unplaced variable palette*, which looks identical to Figure 4, but without the Make Alias button. It contains the variables of the code (in the group hierarchy as the code author defined) that have not yet been placed on any screen in the study. This list gives the study author quick access to changing the defaults for invisible variables by selecting their name from this list to update their properties in the Property Inspector, as well as quick access to

3. See later section for description of layout engines.

placing them on the screen through drag and drop.

The next type of palette is the *screen hierarchy palette*. This palette, shown in Figure 5, can occur multiple times. At the top of the palette is a popup menu showing which study's screens are being listed in the tree control below it. When it says "current", it shows the list of screens for the study being currently worked on. It also contains a list of all of the other studies for the particular code being used. If one of them is selected, the tree will contain the list of screens for that other study previously created for the code. Beneath the list there is the ability to find a screen based on its title as well as the ability to create a new group. The screens have the ability to be structured into groups in the same fashion as the variables when codes are being created. They can also be reorganized between the groups in the same way. By clicking on a screen to select it from the tree, the Property Editor can be used to change the help string and the screen title. This can only be done for screens in the "current" screen list palette for the study being edited. If the user drags a screen from the "current" list into the design area, a screen link button will appear on that screen being edited that links it to the screen that was dropped. If you drop a screen from a list of screens from a different study, that screen will first be copied into the "current" study and then be linked. If you double click a name in the "current" list, the design area will shift to edit that screen. If you double click a name in the list for a different study, you will be prompted to see if you want to copy the screen. Dropping a screen name dragged from a list of a different study onto the list of the "current" study will cause that screen to be copied into the "current" study being edited. Multiple instances of this palette can be created through menu commands.

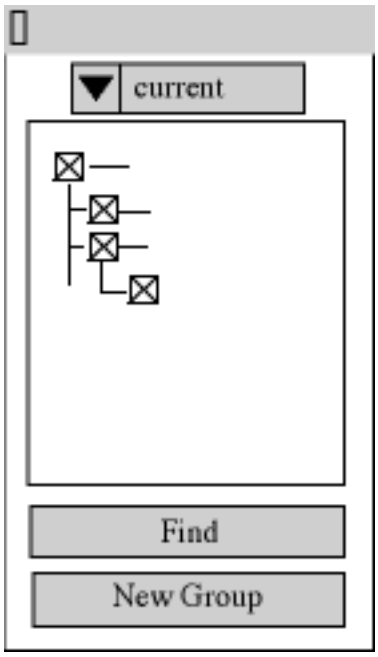

Figure 5: Screen List

Figure 6

;The next type of palette is the *tools palette*, shown in Figure 6. The tools palette allows the study author to choose which tool they want to use in the design area. The tools provided include the arrow for selecting, moving, and resizing objects, a text tool for typing static text, a button tool for linking to other screens, a rectangle tool for drawing rectangles, and a line tool for drawing straight lines.

The *property editor* is a window that is by default empty. The contents of the property editor change depending on which object is currently selected. If the user last clicked on a variable from one of the variable lists, the default value, consistency checks, and one-liner for that variable can be changed. If the user last clicked on a screen from one of the screen list palettes, the user can then change the screen's title and help string. If the user last clicked on a variable input area inside of the design area, the user can change the default value, consistency checks, and one-liner for the variable along with adjusting its appearance on the screen including properties of the javabean.

The study author uses these palettes in conjunction with wizards to author the graphical interfaces of studies. In the menus along with these include an import command for setting default values for variables from a namelist, including any ailiased variables. The study author can also choose to use a wizard that takes lists and groups of variables and automatically constructs screens using the layout editors. Studies can also be created by cloning an existing run and using its variable values as the study defaults.

Standard Variable Appearance

Each instance of a variable on a screen will have a standard interface that is present regardless of any auxiliary information that may be present on the screen. There will be the variable's name, the JavaBean used to edit variables of that type, along with a popup menu button to the left of the variable name. When the user moves the mouse over the variable name, a small Balloon Help style slip will popup and display the variable's one line description. Underneath the popup menu will be more specific commands that relate to the variable including the ability to open up the URL of the extended documentation in the browser, revert the variable to the default value for the study, get the value of a variable from a particular run, and get the value of the variable for a particular namelist. Other interface elements may appear on the screen for different layouts (see next section), but this succinct interface will be the one consistently available for the user.

Layout Engines

The graphical user interface components of UniVista will have different built in layout engines. A *layout engine* is a piece of code that is responsible for taking a list of variables and their groups and creating a singe screen or group of screens that lay them out in a programmatic fashion. They are similar to utilizing templates in a presentation program. The idea behind layout engines is that if a new layout format is desired, someone simply needs to code a new layout engine and it will appear anywhere in the program. There will be three default layout engines available in the program: verbose, sparse, and terse.

The *verbose* layout engine will display the most controls on the screen per variable. For each variable it will attempt to put on the screen the popup menu button, variable name, the entry area, the one-liner, a button to restore it to the study default, and a button linking to the external documentation. Each variable passed to the engine is laid out one underneath the other in a list fashion.

The *sparse* mode lays out variables one underneath each other including the variable name, popup menu button, entry area, and the one-liner. If there is enough room horizontally on the screen, variables will be laid out in two columns.

The *terse* mode lays out variables in one (or possibly two) columns using the standard layout of the popup menu button, the variable name, and entry area.

The layout engines are used in the study editor automatic screen creation wizards, and offer the potential to allow screens to be dynamically laid out during a user's run to allow them to use the style of interface they are most familiar with.

Run Management

The run editor has the ability to create a session from one of the existing codes and studies and modify the values of the variables for a particular run of the simulation software. Runs can be created from a study, or by cloning an existing run. Runs can also be created by specifying a study and a namelist which is used to import values for the variables. Each run is stored in the database underlying UniVista in order to archive each simulation. Variable values and output files are stored. When each run is archived, the person who executed the run can determine whether all users should be able to see the run or whether it should be accessible to that user only.